EMBARRASSINGLY EASY EMBARRASSINGLY PARALLEL PROCESSING IN R: SUPPLEMENTAL APPENDIX

MICHAEL S. DELGADO AND CHRISTOPHER F. PARMETER

This appendix reports the computation time for a standard wild bootstrap used to compute standard errors in a linear model with heteroskedastic errors, nonlinear optimization to solve a nonlinear least squares regression problem, and a Monte Carlo simulation to assess the size of a Hausman (1978) test using `multicore`, `snow`, and `snowfall`. Simulations consider variable number of processors - $\{1, 2, 4, 8, 16, 32, 64\}$. All computations were conducted using a 660 node Red Hat Enterprise Linux 6 (RHEL6) research cluster with dual 8-core Intel Xeon-E5 nodes. Detailed code to reproduce each computation are available in the *JAE* Data Archive.

The tables below report the computation time for each procedure for variable number of processors, for each package. It is clear that there are sizable gains from parallel implementation, but that the marginal time increases diminish as the size of the cluster grows. In each of our examples below, it is clear that implementing a parallel environment above 16 processors leads to virtually no improvement in computation time, and in several cases, leads to slower computation time. This results for two reasons.

First, it is well-documented in computer science that increasing the number of processors does not always lead to increased computational efficiency, as the processors must communicate with each other. This phenomenon is known as 'overhead' - eventually the overhead becomes so great that any additional computational advantages from employing additional processors is dominated by overhead communication so computation time either does not decrease, or may increase.

Second, in our case, the Linux cluster on which these simulations were computed is a system of dual 8-processor nodes. Parallel environments up to and including 16 processors can all be run on one node; environments calling 32 or 64 processors must involve multiple nodes. Hence, overhead greatly increases when connecting multiple nodes, so computational efficiency significantly decreases in these examples as overhead costs substantially increase.

It is also apparent that in a few cases for the bootstrap example, increasing the size of the parallel environment improves computation time by more than 50 percent. Specifically, using `multicore` increasing from 2 to 4 processors decreases computation time by about 51 percent,

PURDUE UNIVERSITY AND UNIVERSITY OF MIAMI

Date: July 27, 2013.

Michael S. Delgado, Department of Agricultural Economics, Purdue University, West Lafayette, IN 47907-2056. Phone: 765-494-4211, Fax: 765-494-9176, E-mail: delgado2@purdue.edu.
Christopher F. Parmeter, Department of Economics, University of Miami, Coral Gables, FL 33124-6520. Phone: 305-284-4397, Fax: 305-284-2985, E-mail: cparmeter@bus.miami.edu.

* This research was supported in part by computational resources provided by Information Technology at Purdue - Rosen Center for Advanced Computing, Purdue University, West Lafayette, Indiana.

and going from 1 to 2 and 2 to 4 processors using `snow` decreases computation time by about 62 and 55 percent. This phenomenon is known as superlinear speedup (Janßen 1987), and may arise in memory intensive computational problems as a parallel environment can sometimes more efficiently use memory relative to the sequential (or smaller parallel) environment. In our bootstrap example, we consider $n = 30,000$ observations with $B = 100,000$ bootstrap replications, which is relatively more memory intensive than our nonlinear optimization and Monte Carlo. Further, the cluster computing environment used for these exercises is a multi-user research cluster. Jobs are automatically allocated to available resources, however allocation does not guarantee that the node deployed for computation is otherwise idle. This potentially adds some slight variability in timing across each exercise as the available amounts of memory across different nodes may not be the same.

References

Hausman, J. A. (1978), 'Specification tests in econometrics', *Econometrica* **46**, 1251–1271.
Janßen, R. (1987), 'A note on superlinear speedup', *Parallel Computing* **4**, 211–213.

Table 1. Computation time using `multicore`.

	Number of Processors						
Model	1	2	4	8	16	32	64
Bootstrap	9578.983	5794.143	2819.724	1601.787	944.875	942.551	1033.977
Optimization	51.731	27.489	13.971	8.328	4.420	4.240	3.685
Monte Carlo	530.164	266.700	152.062	79.875	40.271	40.556	43.349

Table 2. Computation time using `snow`.

	Number of Processors						
Model	1	2	4	8	16	32	64
Bootstrap	9417.737	3576.098	1610.047	1194.447	684.464	750.852	742.129
Optimization	51.410	26.364	15.049	8.735	4.424	3.976	3.976
Monte Carlo	456.730	265.031	140.191	75.794	41.951	42.133	43.472

Table 3. Computation time using `snowfall`.

	Number of Processors						
Model	1	2	4	8	16	32	64
Bootstrap	8169.750	4029.330	2018.471	1207.365	713.637	712.574	709.432
Optimization	53.408	27.505	15.321	8.443	4.458	4.043	4.101
Monte Carlo	486.301	257.592	144.018	76.734	40.451	41.239	45.072